PMX1C Tile Accelerator Software Interface

Document ID	:	1CTASOF
Issue Number	:	Draft A
Issue Date	:	Not Issued
Author	:	Stuart Smith

DOCUMENT HISTORY

Issue	Date	Changes / Comments
Draft A		Not Issued
Draft B		Updated data structures
Draft C	10/5/98	Removed seed macro tiling

BIBLIOGRAPHY

[1] PMX1C Tile Accelerator Specification	Andrew Webber	HTILE02D.DOC
[2] PMX1C Tile Accelerator Software Usage	John Howson	SOFTTILE.DOC

TABLE OF CONTENTS

1. INTRODUCTION

PowerVR2 (PVR2) is a deferred rendering device using tile based display lists. The render surface is broken down into 32x16 tiles (or regions) each one having its own display list containing the data for the objects within that tile. Therefore, before a render can start, the object data for the whole scene must be 'tiled', i.e. a bounding box for each object must be calculated and then the object inserted into the display lists for the tiles that overlap the box. This task is performed by the tile accelerator which can work in one of four different modes, each mode having slightly different input data and advantages/disadvantages. To achieve some of the more complex visual effects seen in 3D scenes, the PVR2 hardware supports multiple passes on each tile. To support this the tile accelerator must correctly handle multiple passes through the use of multiple region headers.

1.1 Tile Accelerator Modes

The tile accelerator's different modes are intended to provide optimum performance across a range of PMX interface and memory options. Copy Through will undoubtedly be the fastest mode on AGP systems but will carry the burden of requiring a significant buffer of video memory. If a buffer of required size is unavailable then a Macro Tile mode will become the preferred mode of operation. Full Macro Tiling may be more optimal on PCI systems as the amount of data refetches over the system bus is minimised. It is unlikely that Non-copythrough mode will be used. Driver logic will attempt to select the optimum mode practical for a given application.

1.1.1 Copy Through

In Copy Through mode the TA reads a control stream consisting of ISP/TSP control words followed by object pointers for all objects that share those words. The TA reads the raw object data the pointers reference from Host memory, applies small object culling, applies scaling to X and Y co-ordinates if super sampling is enabled and inserts the ISP and TSP control words and writes the resulting object data into frame buffer.

The tile accelerator is started by requesting a tile accelerator interface from PMXDXSRV and then writing the start command into the circular command buffer of the primary core. In Copy Through mode this is done once per render after the object and object pointer data for the entire scene has been stored

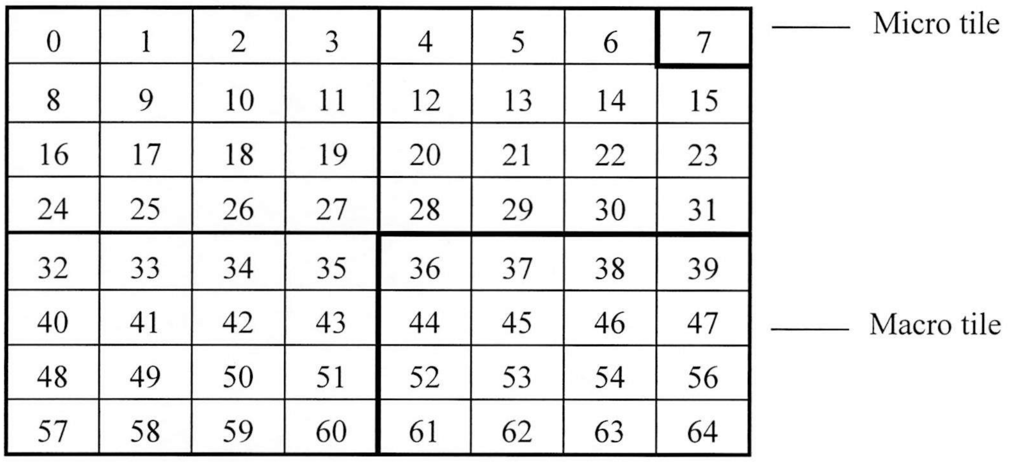
1.1.2 Non-Copy Through

In Non-Copy Through mode the ISP/TSP control words are in the object data buffer, each object having its own set of control words. The disadvantage of this mode is that the repeated control words waste bandwidth when being read from the host.

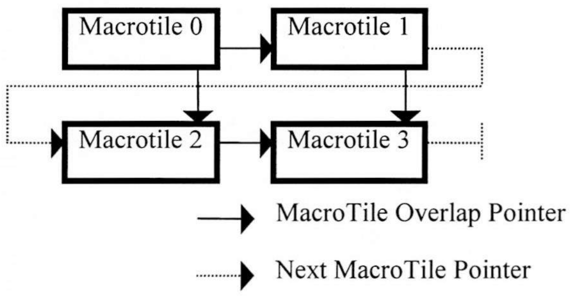
The Host driver can kick off the tile accelerator each time its has collected a sufficient number of object pointers to avoid the tile accelerator start-up overhead impacting performance.

1.1.3 Full Macro Tiling

A macro tile is a tile that is composed of XxY micro tiles (PVR2 internal tiles i.e. 32x16). A render surface can be viewed as a single macro tile i.e. simple Copy Through mode or as a number of smaller macro tiles. The later would generally take the form of quarters or sixteenths in order minimise the cost of Host tiling. By breaking the scene data down into large sub units the number of times data is read from host memory can be kept to a minimum whilst minimising the amount of frame buffer required to store the parameters.



For full macro tiling the Host driver needs to supply the object pointer to all macro tiles that overlap the object's bounding box.



The tile accelerator is only started once per render with the control list in Copy Through mode format (Copy Through mode is just macro tiling with a single device-sized macro tile).

1.2 Multiple Region Headers

Each region allows four hardware passes per screen region (or tile), these being split into, opaque objects, opaque volumes, translucent objects and translucent volumes. If all objects were to be inserted into one region this approach would severely limit the performance and functionality of visual effects such as tri-linear mip-mapping, multi-layered texturing and effect volumes. In addition, unacceptable limitations would be placed on the combinations of translucent objects with ordering dependencies.

To overcome these limitations multiple region headers (or passes) can be defined per region. Each region header can specify:

- Auto or pre-sorted translucency mode,
- If ISP depth should be cleared,
- If accumulation buffer should be flushed to frame buffer at end of pass.

The following sections are examples of the cases where multiple region headers will give improved performance per render output not achievable with one region header.

1.2.1 Trilinear

Trilinear filtering can be implemented by using a number of region headers. E.g. Opaque trilinear filtering requires two opaque pass/list blocks (see Chapter 3: Data Format) with the same object data pointed to by 2 object pointers (one in each block). The pointer in the first block has control words specifying multipass-texture pass A with a blend of 1 0 into the primary accumulation buffer. The second pointer has control words specifying multipass-texture pass B with a blend of 1 1 into the primary accumulation buffer. The pass flags must be set to retain z-state and to not flush the accumulations buffer.

Translucent trilinear requires three translucent pass/list blocks. The control words for the object in the first block do a blend of 1 0 into the secondary accumulation buffer. The object's control words in the second block specify a 1 1 blend into the secondary accumulation buffer, and the control words in the third block specify a 1 1 blend into the primary accumulation buffer.

1.2.2 Multi-texturing

Multi-texturing is performed by placing each texture layer in a different region header. E.g. for a three layer translucent multi-textured object 4 translucent pass/list blocks are needed. As with trilinear filtering the same object data is pointed to by pointers each having different associated control words.

A three layer opaque multi-textured object requires only three pass/list blocks as the final blend with the primary accumulation buffer is 1 0.

1.2.3 Punch Through Translucency

In a conventional Z buffer based system punch through translucency doesn't need to be sorted by the application. As PVR uses deferred texturing (with no feed back of CK/Alpha test to ISP), all punch through textured polygons need to go through a fully sort translucent pass. As the majority of titles are written to run on other platforms besides PVR (particularly D3D apps) they tend to assume that translucent data has the same full set of Z compare modes as opaque data. This precludes the use of full-sort translucency mode all the time.

PMX1C allows punch through translucency to be grouped into a separate pass that has full translucency sorting enabled, so removing the Z Mode restrictions from the true translucent data.

In addition to this it is possible to support "layered" punch through sorting i.e. split the punch through data into N passes each covering fixed Z ranges. Objects should be placed into passes based on their back-most Z co-ordinate. In a scene with a uniform spread of punch-through data across the valid Z range this processing could massively reduce the effect of n^2 sorting.

1.2.4 Cheap and Flexible Volume Effects

One of the primary criticism's of PVR2's modifier volumes is that they are extremely expensive to use for even simple effects such as shadow and light volumes as double the amount of material data has to be supplied.

PMX1C can overcome this by processing the required "reactive" effect as a separate single full-screen object. This is achieved by sending all scene data and then starting a new region header (retain Z and accumulation buffer state) that contains the effect object and the volumes to which you wish to apply that effect. The effect object MUST have its depth compare mode set to always and Depth Write disabled, this has the effect of replacing the front most ISP object TAG with that of the effect object without destroying the depth value. When the hardware processes the volumes contained with the pass (region header) the TSP will simply apply the material and blending of the effect object based on the inside outside information to the current contents of the accumulation buffer.

The additional benefit is that any number of differing effects volumes can be applied across the scene e.g. true interacting light and/or shadow volumes, multiple concentric volumes to simulate soft edged effects etc.

2. DATA FORMAT

Before starting the tile accelerator, the following data structures must be set up. Full macro tiling mode uses a list of macro tile blocks (that must be contained within one physical page).

2.1 Macro Tile block

DWORD TA Inst 2 (low DWORD)
DWORD TA Inst 2 (high DWORD)
DWORD Right Macrotile Reference Pointer
DWORD Below Macrotile Reference Pointer
DWORD Log2NumPasses
DWORD MacroTileW*MacroTileH*4

The **Right Macrotile Reference Pointer** must be set to NULL if only one macro tile exists or if the macro tile is bottom-right corner of the screen.

The **Below Macrotile Reference Pointer** must be set to NULL if only one macro tile exists or if the macro tile is at the bottom of the screen.

Each macro tile block is followed by a number of pass/list blocks that describe the passes (region headers) and lists for that macro tile. The total number of region headers following the macro tile block is specified in **Log2NumPasses**.

2.2 Pass/List block

DWORD TA Inst 1 (high DWORD)
DWORD TA Inst 3 (high DWORD)
DWORD Num TA Ctrl List DWORDS
DWORD TA Ctrl List Page Table Pointer

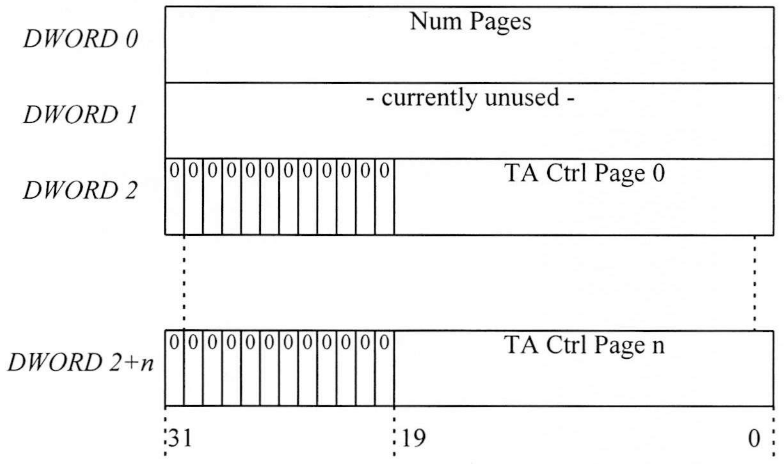
If seed macro tiling is being used, then all macro tiles must use the same pass relationship. That is, if tile A has 3 passes of meaning X, Y and Z then all subsequent tiles must contain the same passes, in the same order and position, even if they don't require them. This is so that the primary core software can place overlapping objects into the correct pass position. Any unused passes must have a Ctrl List DWORD count of zero.

The pass/list array is terminated with the top-bit in the first word set to '1' (i.e. see Inst) in the last pass/list block. This is because the number of pass/list blocks depends not only upon the number of passes but also upon the lists used within those passes.

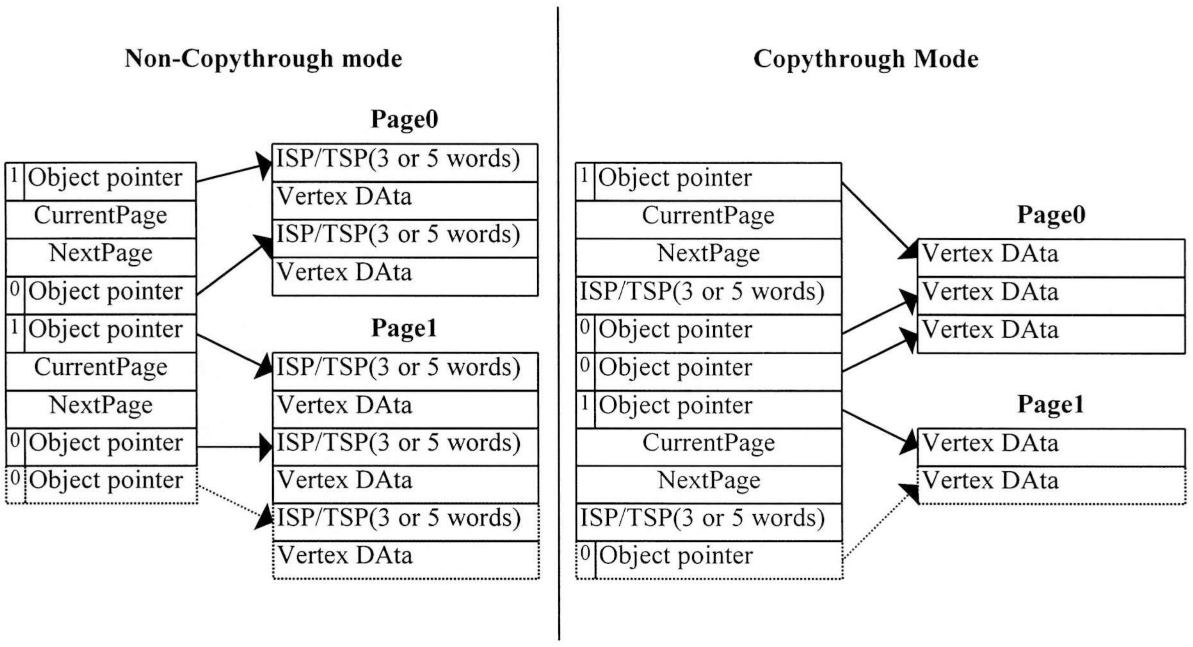
Inst 1 also defines information such as the pass and list number of the current pass/list block. The stride and offset values are used to interleave translucent objects (List 2) for trilinear and multi texturing. If the list is not translucent then the offset should be set to 0 and the stride to 1. To implement translucent trilinear 3 pass/list descriptor blocks are required. The first block has a stride and offset of 3 and 0; second 3 and 1; third 3 and 2.

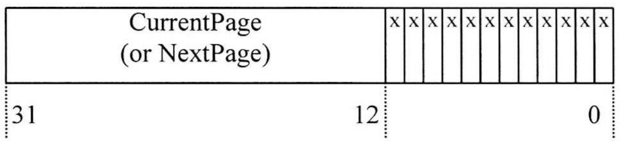
2.3 Page table

The page table is an array of control-list page addresses with the first DWORD in the table used as a 'number of entries' count.


DWORD 0
Num Pages
DWORD 1
- currently unused -
DWORD 2
0 0 0 0 0 0 0 0 0 0 0 0
TA Ctrl Page 0
DWORD 2+n
0 0 0 0 0 0 0 0 0 0 0 0
TA Ctrl Page n
31
19
0

2.4 Control List format





The control list buffer contains pointers to the objects in the scene and some control data. The arrangement and type of the control data depends on whether the tile accelerator is in Copy Through, Non-Copy Through, or macro tiled mode.

The pointers are in internal PVR2 parameter format:

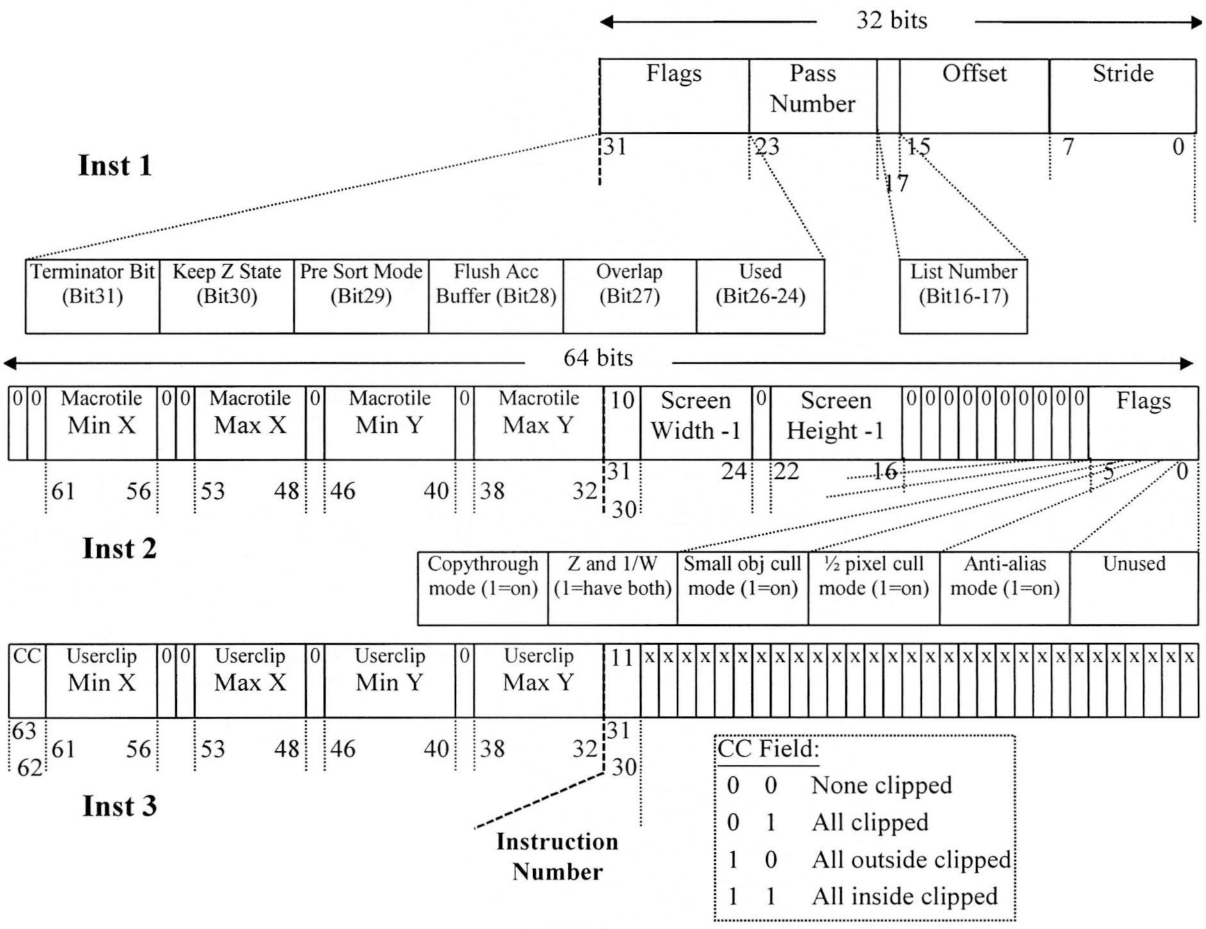
Bits 31	*30-25*						*24*	*23-21*	*20-0*
X	Mask						Shadow	Skip	Triangle Strip Start (32bit Word Address)
	T0	T1	T2	T3	T4	T5			

In Copy Through/macro tiled modes the pointer buffer also contains the ISP and TSP control words. These are inserted in addition to the current and next physical page pointers after the escape pointer value.

As PMX1C has no address LUT the object pointer lists must also contain the physical page bases for all referenced object data. Whenever an object overlaps a page boundary an escape pointer must be inserted that indicated that the new current and next physical page addresses follow. In Copy Through mode an escape pointer is also needed when the control words of an object differ from those of the previous. The escape pointer is simply a valid object pointer with its MSB set (new pages apply to this and all subsequent pointers).

Note: The escape pointer block (escape pointer and ISP/TSP control words) must not cross the page boundary between the fourth and fifth physical pages of a tile accelerator control list.

2.5 Tile Accelerator Instructions



3. Examples

3.1 Rendering strips in Copy Through mode

1 Create a single macro tile (with its size set to the size of the screen device) and a number of pass/list blocks. The number of blocks depends on the type of operations required e.g. one block is needed per multi-texture layer.

2 Split the strips into strips of maximum length 6 (as maximum length of hardware strip is 6 triangles) and pack the vertex data (in PVR2 data format) into the object data buffer.

3 Calculate object pointers for the packed object data and store them in a temporary buffer. A temporary buffer is used to prevent the performance hit that occurs in AGP systems when writes to different buffers are interleaved.

4 Insert the object pointers into the tile accelerator control lists:

```
While (pointers left)
{
      if (not enough space left in current TA Ctrl List block)
      {
            Create new TA Ctrl List block and add to page table
      }

      write first pointer as escape pointer
      write currentpage, nextpage and associated control words

      pointers left— -

      while (pointers left
            && space left in current TA Ctrl List block
            && object has same control words as previous)
      {
            write pointer
            pointers left --
      }
}
```

5 Start tile accelerator by obtaining a tile accelerator interface structure from PMXDXSRV and issuing a start command.

3.2 Rendering strips in Full Macro Tile mode

1 Create a NxM macro tiles (dividing the screen device into tiles whose widths are multiples of 32 pixels and whose heights are multiples of 16 pixels) and a number of pass/list blocks for each macro tile. The number of blocks (which must be the same for each tile) depends on the type of operations required e.g. one block is needed per multi-texture layer.

2 Split the strips into strips of maximum length 6 (as maximum length of hardware strip is 6 triangles) and pack the vertex data (in PVR2 data format) into the object data buffer.

3 Calculate object pointers for the packed object data and store them in a temporary buffer.

4 Calculate the maximum and minimum X and Y co-ordinates for all the vertices in each of the strips, and use these values to determine the macro tiles that each object overlaps.

5 Insert the object pointers into the control lists of the seeded tiles using a similar method to 5) in section 4.1

6 Start tile accelerator by obtaining a tile accelerator interface structure from PMXDXSRV and issuing a start command.